3PixelRule_Gray.tiff ,

SilkScreen v1.0

An IconBuilder Filter
Toby Paterson
paterson@gdss.commerce.ubc.ca

v1.0 March, 1993

131068_PixelRule_Gray.tiff ,

The SilkScreen filter

In IconBuilder, a filter is an operation which is invoked on the current selection. The two default filters are Flip and Rotate, which do the obvious things to the selection, and IconBuilder provides the ability to dynamically load new filters (and new tools for that matter). In the Fall 1991 issue of **Next On Campus**, a copy of which can be found sonata.cc.purdue.edu in `/pub/next/Newsletters/OnCampus/NOCFall91`, the article *An Introduction to Extensible Programming* by Jeff Martin and Joshua Doenias describes the API for a program called DynaPaint; this is the same API that is used in IconBuilder.

SilkScreen is a filter which changes pixels of one colour to another: all pixels in the selection are scanned, and any pixels which are sufficiently close to the source colour are replaced by the destination colour. I use it for those cases where the fill operation isn't sufficient (on dithered colours, for example) or when there are a few pixels whose colour I want to change and which are scattered throughout the image.

How to load the filter

Start up IconBuilder and load the SilkScreen filter using the `Tools/Load Tool` menu item. Two things should be noted:

- double clicking on the filter from the Workspace browser, although it launches IconBuilder, fails to load the filter correctly (at all!).
- the filter must be loaded before the selection tool's inspector is displayed for the first time.

Alternatively, you can place the SilkScreen.pfilter bundle into the IconBuilder app wrapper and it will automatically be loaded when IconBuilder is launched.

The SilkScreen filter operates on the current selection. Pick the selection tool, select the portion of the image you want to mung, and open the inspector window (from the `Tools/Inspector...` menu item or cmd-j). Drag the pop-up menu to SilkScreen, set the attributes as you want them (see below) and hit the Apply button.

Using the SilkScreen filter

The SilkScreen filter scans all pixels in the current selection and changes

the colours of those pixels which match the source colour to the destination colour. By default, the colour comparison between the source colour and the colour of the pixels is performed in the RGBA colour space. The Euclidean distance between the two colours is computed and if it is less than the specified tolerance, the pixel's colour is changed to the destination colour.

The tolerance is controlled by a slider in the inspector window and is given as a percentage figure. A 0% tolerance means only those pixels which are exactly the same as the source colour are converted; 100% means all pixels in the selection will match the source colour and hence be converted. (If you picture RGB space as a colour cube, then the tolerance specifies the radius of a sphere centred at the point representing the source colour. All colours which fall within this sphere will match the source colour and be converted to the destination colour.)

foo.eps ,

Using the channel selector buttons, it is possible to limit which channels are used in the comparison. Only those channels which are selected will be used when comparing the two colours for a match. So it is possible, for instance, to compare only the red components or the red and blue components of two colours and ignore the rest.

If the interpolation button is selected, then the destination colour is modified proportionally to reflect the difference between the source colour and the pixel colour. For example, if the pixel colour contains more red than the source colour (but still falls within the tolerance, of course), then the new pixel colour will be the destination colour with proportionally more

red added to it. This comparison and modification is performed in the HSB colour space, all components of the colour space are used and you have no choice in this matter.

Bugs/Things it would be nice to have

- it would be nice to have a choice in this matter
- comments in the code
- provide the ability to compare the source and pixel colours for a match in different colour spaces
- show a colour wheel graphically displaying the tolerance factor and which colours will match the source colour
- I haven't tested the bitmap ReadPixel()/WritePixel() routines on all types of images; if you come across a format which causes the code to barf, please let me know (and mail me the fixes too if you like :)